DOCUMENT NEAR-DUPLICATE DETECTION

BACKGROUND OF THE INVENTION

A. Field of the Invention

[0001] Systems and methods consistent with the principles of the invention relate generally to document processing and, more particularly, to comparing documents to find near duplicate documents.

B. Description of Related Art

[0002] There are a number of applications in which it may be desirable to determine whether documents are near duplicates of one another. In the context of the World Wide Web, for example, search engines typically provide a searchable index of numerous web pages. Frequently, web pages located at different locations may be duplicates or near duplicates of one another. Knowing when one web page is a near-duplicate of another can be beneficial both when archiving the web pages and when returning search results to a user in response to a search query.

[0003] An archive server, for example, may be designed to store an archive of all documents requested from a web server. The archive server may decide whether to store new incoming documents based on whether the new document is a duplicate or near-duplicate of a previously stored document.

[0004] Thus, there is a need in the art for accurate and efficient techniques for automatically detecting near-duplicate documents.

SUMMARY OF THE INVENTION

[0005] A method consistent with an aspect of the invention generates a representation of a document. The method includes sampling the document to obtain overlapping blocks, choosing a subset of the sampled blocks, and compacting the subset of the sampled blocks to a obtain the representation of the document.

[0006] Another method consistent with an aspect of the invention generates a representation of a document. The method includes sampling the document to obtain overlapping samples and selecting a predetermined number of the samples as those of the samples corresponding to a predetermined number of smallest samples or a predetermined number of largest samples. The method further includes setting bits in the representation of the document based on the selected predetermined number of the samples.

[0007] A device consistent with an aspect of the invention includes a fingerprint creation component and a similarity detection component. The fingerprint creation component generates a fingerprint of a predetermined length for an input document. The fingerprint is generated by sampling the input document, choosing a subset of the samples, and generating the fingerprint from the subset of the samples. The similarity detection component compare pairs of fingerprints to determine whether the pairs of fingerprints correspond to near-duplicate documents.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and, together with the description, explain the invention. In the drawings,

[0009] Fig. 1 is a diagram illustrating an exemplary overview of document near-duplicate detection;

[0010] Fig. 2 is an exemplary diagram of a network in which systems and methods consistent with the principles of the invention may be implemented;

[0011] Fig. 3 is an exemplary diagram of a client or server according to one embodiment of the invention;

[0012] Fig. 4 is a block diagram illustrating functional components of the near-duplicate component shown in Fig. 2;

[0013] Fig. 5 is a flow chart illustrating operations consistent with one embodiment for sampling a document during fingerprint generation;

[0014] Fig. 6 is a diagram illustrating application of a fixed-sized sliding window;

[0015] Fig. 7 is a flow chart illustrating operations consistent with an embodiment for compacting;

[0016] Fig. 8 is a diagram conceptually illustrating acts performed with respect to Fig. 7; and

[0017] Fig. 9 is a diagram illustrating an exemplary implementation of the near-duplicate component in the context of an Internet search engine.

DETAILED DESCRIPTION

[0018] The following detailed description of the invention refers to the accompanying drawings. The detailed description does not limit the invention.

OVERVIEW

[0019] As described herein, a near-duplicate component includes a fingerprint creation component and a similarity detection component. The fingerprint creation component receives a document of arbitrary size and generates a compact "fingerprint" that describes the contents of the document. Because the file size of the fingerprint is relatively small, it can be efficiently stored and retrieved from computer memory. Fingerprints from other documents can also be stored and easily and efficiently compared to one another by the similarity detection component to determine if the documents are duplicates or near-duplicates of one another.

[0020] Fig. 1 is a diagram illustrating a document and its fingerprint. Document 110 may be a relatively large document (e.g., 100,000 bytes). Fingerprint 120, created from document 110, is much smaller (e.g., 8 or 16 bytes). Despite the size difference, fingerprint 120 can be compared to other fingerprints to determine whether the underlying documents are near-duplicates of one another.

EXEMPLARY NETWORK OVERVIEW

[0021] Fig. 2 is an exemplary diagram of a network 200 in which systems and methods consistent with the principles of the invention may be implemented. Network 200 may include multiple clients 210 connected a server 220 via a network 240. Network 240 may include a local area network (LAN), a wide area

network (WAN), a telephone network, such as the Public Switched Telephone Network (PSTN), an intranet, the Internet, or a combination of networks. Two clients 210 and one server 220 have been illustrated as connected to network 240 for simplicity. In practice, there may be more clients and/or servers. Also, in some instances, a client may perform the functions of a server and a server may perform the functions of a client.

[0022] Clients 210 may include client entities. An entity may be defined as a device, such as a wireless telephone, a personal computer, a personal digital assistant (PDA), a lap top, or another type of computation or communication device, a thread or process running on one of these devices, and/or an object executable by one of these devices. Server 220 may include a server entity that processes, searches, and/or maintains documents in a manner consistent with the principles of the invention. Clients 210 and server 220 may connect to network 240 via wired, wireless, or optical connections.

[0023] In an implementation consistent with the principles of the invention, server 220 may implement near-duplicate component 225. In general, near-duplicate component 225 may receive documents from any of a number of possible sources, such as clients 210, server 220, or other server entities coupled to network 240. Near-duplicate component 225 may generate compact fingerprints for these documents and/or compare fingerprints to determine if two documents are duplicates or near-duplicates of one another.

[0024] A document, as the term is used herein, is to be broadly interpreted to include any machine-readable and machine-storable work product. A document

may be an e-mail, a blog, a file, a combination of files, one or more files with embedded links to other files, a news group posting, etc. In the context of the Internet, a common document is a web page. Web pages often include content and may include embedded information (such as meta information, hyperlinks, etc.) and/or embedded instructions (such as Javascript, etc.).

EXEMPLARY CLIENT/SERVER ARCHITECTURE

[0025] Fig. 3 is an exemplary diagram of a client 210 or server 220 according to an implementation consistent with the principles of the invention. Client/server 210/220 may include a bus 310, a processor 320, a main memory 330, a read only memory (ROM) 340, a storage device 350, an input device 360, one or more an output device 370, and a communication interface 380. Bus 310 may include a set of conductors that permit communication among the components of client/server 210/220.

[0026] Processor 320 may include a conventional processor or microprocessor that interprets and executes instructions. Main memory 330 may include a random access memory (RAM) or another type of dynamic storage device that stores information and instructions for execution by processor 320. ROM 340 may include a conventional ROM device or another type of static storage device that stores static information and instructions for use by processor 320. Storage device 350 may include a magnetic and/or optical recording medium and its corresponding drive.

[0027] Input device 360 may include conventional mechanisms that permit a user to input information to client/server 210/220, such as a keyboard, a mouse, a

pen, voice recognition and/or biometric mechanisms, etc. Output device 370 may include conventional mechanisms that output information to the user, including a display, a printer, a speaker, etc. Communication interface 380 may include any transceiver-like mechanism that enables client/server 210/220 to communicate with other devices and/or systems. For example, communication interface 380 may include mechanisms for communicating with another device or system via a network, such as network 240.

[0028] As will be described in detail below, server 220, consistent with the principles of the invention, may implement near-duplicate component 225. Near-duplicate component 225 may be stored in a computer-readable medium, such as memory 330. A computer-readable medium may be defined as a physical or logical memory device and/or carrier wave.

[0029] The software instructions defining near-duplicate component 225 may be read into memory 330 from another computer-readable medium, such as data storage device 350, or from another device via communication interface 380. The software instructions contained in memory 330 may cause processor 320 to perform processes that will be described later. Alternatively, hardwired circuitry or other logic may be used in place of or in combination with software instructions to implement processes consistent with the present invention. Thus, implementations consistent with the principles of the invention are not limited to any specific combination of hardware circuitry and software.

NEAR-DUPLICATE COMPONENT

[0030] Fig. 4 is a block diagram illustrating functional components of near-duplicate component 225. As shown, near-duplicate component 225 includes a fingerprint creation component 410 and a similarity detection component 420. Fingerprint creation component 410 may receive an input document and generate a fingerprint that is a compact representation of the document. In one implementation, the fingerprints are each 128 bit (16 byte) values.

[0031] Similarity detection component 420 may generate a measure of similarity between two documents based on the fingerprints corresponding to the two documents. In one implementation, the measure of similarity may be computed as the hamming distance between its two input fingerprints. For binary values, the hamming distance can be defined as the number of digit positions in which the corresponding digits of two binary words of the same length are different. For example, the hamming distance between 1011101 and 1001001 is two. For a 128 bit fingerprint, a hamming distance of 18 may be used to distinguish whether two documents are near-duplicates, i.e., a hamming distance less than or equal to 18 can indicate near-duplicate documents, otherwise, the documents are considered non-near-duplicates. One of ordinary skill in the art will recognize that other threshold levels could be used depending on the particular application.

[0032] For larger thresholds, the probability for a false negative is lower but the probability for a false positive is higher. For smaller thresholds, the probability for a false positive is lower but the probability for a false negative is higher. A false negative is defined as two near-duplicate pages whose near-duplicate

fingerprints have a hamming distance bigger than the threshold. A false positive is defined as two random pages whose near-duplicate fingerprints have a hamming distance smaller than the threshold. The choice of the threshold determines the balance between false positives and false negatives. Generally, a good choice for the threshold gives nearly the same probability for false positives and false negatives. However, it will be appreciated that various thresholds and proportions of false negatives or positives may be selected in alternative embodiments of the invention.

FINGERPRINT CREATION COMPONENT

[0033] Two general operations may be performed by fingerprint creation component 410 to generate a fingerprint: sampling and compacting. More specifically, an input document is first sampled to generate a number of sampled blocks. If the two documents have x% difference (where x is generally a relatively small number), the sampled documents will generally have approximately x% difference. The sampled blocks are then compacted to obtain a fingerprint of an intended size. In one embodiment, the fingerprints of the two documents should have less than twice the x% difference.

[0034] Fig. 5 is a flow chart illustrating a method for sampling a document during fingerprint generation, according to one embodiment of the invention. A first sample block from the input document may be obtained from a fixed-size sliding window applied to the document (act 501).

[0035] Fig. 6 is a diagram illustrating application of a fixed-sized sliding window applied to a document. The simple exemplary document shown in Fig. 6 is

document 610, which consists of the sentence "Four score and seven years ago." Assume that a 4 character (byte) block size is to be used. The first sampled block, block 620, includes the first four characters from the document ("Four"). A second sampled block, block 621, may include the second through fifth characters in the document ("our "). The third sampled block, block 622, may include the third through sixth characters in the document ("ur s"). In this manner, the sampled four-character blocks "slide" across the document. At the end of the document, boundary conditions may be handled by wrapping the sampling block back to the beginning of the document. Accordingly, the next-to-last sampled block, block 630, may be "o.Fo" and the last sampled block, block 631, may be ".Fou".

[0036] Although the sampled blocks shown in Fig. 6 are four bytes long, in practice, a longer (or even shorter) sampling size may be used. In one implementation, a 64-byte sampling block is used. When sampling documents smaller than the sampling size (e.g., 64 bytes), null characters may be padded to the end of the document until the document equals the sampling size.

[0037] Referring back to Fig. 5, a checksum may be computed for the first sampled block (act 502). For 64-byte sampled blocks, the checksum may be, for example, a 32-bit (4 byte) checksum. In general, a checksum is a number computed by combining characters from a file using a pre-determined mathematical function. Checksum functions are well known in the art, and a number of different checksum calculation functions may be used to generate the

checksums. Alternatively, a hash function may be applied to the sampled blocks to generate the "checksum" values.

[0038] The operations of sampling and computing appropriate checksum values may be performed for each sampled block of the document (acts 503, 504, and 505). A set of the calculated checksum values may next be selected (act 506). For example, the smallest unique 128 checksums may be chosen. In other possible implementations, the largest unique 128 checksums may be chosen. In general, the chosen checksums will correspond to a series that corresponds to random block samplings from the document but that are predetermined in the sense that duplicate or near-duplicate documents will tend to have checksums chosen that correspond to the same text blocks. This set of checksums (e.g., 128 checksums) functions as an effective digest of the document.

[0039] As previously mentioned, fingerprint creation component 410 generally performs sampling and compacting operations to generate the fingerprints. Figs. 5 and 6 illustrate operations for the sampling. In general, compacting refers to reducing the size of the sampled information to obtain a fingerprint suitable for near-duplicate detection. Fig. 7 is a flow chart illustrating a method for compacting, according to one embodiment of the invention. The compacting is performed on the set of checksums generated in act 506. One of ordinary skill in the art will recognize that other compacting techniques could be used.

[0039] Fingerprint creation component 410 may begin by initializing the fingerprint value to zero (act 701). Fig. 8 is a diagram conceptually illustrating

acts performed Fig. 7. A fingerprint value 810 is shown as a 128 bit value initialized to zero.

[0040] The checksums generated in act 506 (Fig. 5) are used to index bits in fingerprint 810. To this end, the checksums may be reduced in length to a length that addresses fingerprint 810 (act 702). In the exemplary implementation described herein, for a 128 bit fingerprint 810, seven bits are required (2 to the seventh power is 128). Accordingly, each of the checksums may be reduced in length to seven bits. The length reduction can be performed via a hashing algorithm. In one implementation, the hashing algorithm may be implemented by taking the least significant seven bits from each checksum. One of ordinary skill in the art will recognize that other, perhaps more complicated, hashing algorithms could be used.

[0041] In Fig. 8, six exemplary checksum values, 820-825, are shown after being hashed to seven bits. Each one of values 820-825 addresses a bit in fingerprint 810. Multiple ones of values 820-825 may address the same bit in fingerprint 810. As shown, values 820 and 821 address the same bit (bit zero) and values 823-825 address the same bit (bit 126). Value 822 addresses bit one. Some bits in fingerprint 810 may not be addressed.

[0042] The bit addressed by a hashed version of a checksum value is flipped each time it is addressed (act 703). Fingerprint 830 illustrates the fingerprint after bit flipping. Bit zero of fingerprint 830 was flipped by value 820 and then flipped back by value 821 and has a final bit value of zero. Bit one was flipped by

value 822 for a flipped value of one. Bit 126 was flipped three times, once by each of values 823-825, resulting in a final bit value of one.

[0043] After bit flipping, the final fingerprint 830 provides a compact representation of the input document for near-duplicate detection.

EXEMPLARY IMPLEMENTATION

[0044] Fig. 9 is a diagram illustrating an exemplary implementation of near-duplicate component 225 in the context of an Internet search engine. A number of users 905 may connect to a search engine 910 over a network 915, such as the Internet. Search engine 910 may be a traditional search engine that returns a ranked set of documents related to a user query. Search engine 910 may be a general search engine, such as one based on all documents from a large collection, such as documents on the web, or a more specialized search engine, such as a news search engine. In other implementations, search engine 910 may be implemented over a specialized corpus of documents, such as a corporate document database made available over a corporate network 915.

[0045] In operation, search engine 910 may receive a user query and generate a list of documents (search results) that contain the terms of the user query. Near-duplicate component 225 may be used by search engine 910 when indexing documents. For example, search engine 910 may use near-duplicate component 225 to avoid redundantly indexing/storing multiple versions of a same or very similar document.

[0046] Near-duplicate component 225 can be used in applications other than assisting search engines. For example, the nearest duplicates of two documents can be used as a base for differential compression.

CONCLUSION

[0047] Techniques for efficiently representing documents for near-duplicate detection were described with reference to exemplary embodiments of the invention.

[0048] It will be apparent to one of ordinary skill in the art that aspects of the invention, as described above, may be implemented in many different forms of software, firmware, and hardware in the implementations illustrated in the figures. The actual software code or specialized control hardware used to implement aspects consistent with the present invention is not limiting of the present invention. Thus, the operation and behavior of the aspects were described without reference to the specific software code -- it being understood that a person of ordinary skill in the art would be able to design software and control hardware to implement the aspects based on the description herein.

[0049] The foregoing description of exemplary embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention. For example, although many of the operations described above were described in a particular order, many of the operations are amenable to being performed simultaneously or in different orders to still achieve

the same or equivalent results. Additionally, although primarily described in the context of web sites on the Internet, the concepts discussed above could be applied to other applications.

[0050] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items.